Elective in Robotics

Monocular Visual Odometry
(based on Luca Ricci master thesis)
Monocular vs. Stereo: examples from nature

**predator**
- predators’ eyes face forward
- the field of view of each eye overlaps to create binocular vision (**stereo vision**)

**prey**
- preys’ eyes that sideways
- only a small overlap area between each eye field of view (**monocular vision**)

prey

Predator

Prey
### Monocular vs. Stereo: examples from nature

**Monocular vision features:**
- increased field of view (FOV)
- limited depth perception

<table>
<thead>
<tr>
<th>vision</th>
<th>monocular</th>
<th>stereoscopic</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOV</td>
<td>wide</td>
<td>narrow</td>
</tr>
<tr>
<td>eyes FOV overlapping</td>
<td>does not overlap</td>
<td>overlapping</td>
</tr>
<tr>
<td>estimation of distances</td>
<td>not accurate</td>
<td>accurate</td>
</tr>
</tbody>
</table>
Monocular vs. Stereo: recovering depth

Nature

Pigeons head bobbing generates parallax motion for depth perception.

Geometry

The geometric relationship between the projections of the same point in 2 different views allows depth computation.

Computer Vision

Parallel Tracking and Mapping (PTAM) uses geometry to recover depth and provides camera pose estimation through monocular vision.
Epipolar Geometry

\[ \lambda_1 x_1 = X_1 \]

\[ \lambda_2 x_2 = X_2 \]

\[
\begin{align*}
X_2 &= RX_1 + T \\
&\text{or} \\
\lambda_2 x_2 &= R\lambda_1 x_1 + T
\end{align*}
\]

\( \lambda_1, \lambda_2: \) feature depths in \( O_1, O_2; \)
\( x_1, x_2: \) feature image coordinates in \( O_1, O_2 \)
The Epipolar Constraint

1. \( \lambda_2 x_2 = R \lambda_1 x_1 + T \)

2. \( \lambda_2 \hat{T} x_2 = \hat{T} R \lambda_1 x_1 \)

**elimination of depth**

**epipolar constraint**

\[ x_2^T \hat{T} R x_1 = 0 \]

**essential matrix**

\[ E = \hat{T} R \]

**epipolar geometry entities**

\((O_1, O_2, X) \rightarrow\) epipolar plane

\(O_1, O_2 \rightarrow\) baseline

\(l_1, l_2 \rightarrow\) epipolar lines

\(e_1, e_2 \rightarrow\) epipoles
The Essential Matrix

- A special 3 x 3 matrix encoding epipolar geometry of two views

\[ E = \begin{cases} \hat{R} \mid R \in SO(3), T \in \mathbb{R}^3 \end{cases} \]

- Apparently 8 dof (9 matrix elements up to scale)
- Practically 5 dof (3 – rotation, 2 – translation up to scale)
- Given a point in one image, multiplying by the essential matrix will provide the epipolar line to search along in the second view

Estimation of the essential matrix from a pair of views

(8 – point algorithm | 5 – point algorithm)

1. Rewrite


\[ a = [x_1x_2, x_1x_2, x_1x_2, x_1x_2, x_1x_2, x_1x_2, x_1x_2, x_1x_2, x_1x_2]^T \]

2. Collect epipolar constraints

\[ \chi E^S = 0 \]

\[ \chi = [a^1, \ldots, a^n]^T \]

For uniqueness of solution, \( \chi \) must have at least rank 5

Need parallax motion

\( T = 0 \) won’t work!
## Monocular Visual Odometry vs Monocular Visual SLAM

<table>
<thead>
<tr>
<th>Monocular Visual Odometry</th>
<th>Monocular Visual SLAM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>feature matching between image frame</strong></td>
<td><strong>feature matching between current image frames and a live map</strong></td>
</tr>
<tr>
<td>• faster: works in constant time</td>
<td>• slower but accurate</td>
</tr>
<tr>
<td>• accumulated small errors will cause drifts</td>
<td>• repeated observation of the same features ensures no drifts in trajectory estimate</td>
</tr>
<tr>
<td>• cannot maintain a consistent scale from couples of frames (need 3 or more views)</td>
<td>• scale fixed once set the map</td>
</tr>
<tr>
<td>• motion singularities (pure rotations do not constraint enough the motion)</td>
<td>• extra cost for expanding and maintaining the map</td>
</tr>
<tr>
<td></td>
<td>• method based on EKF are limited by the size of the map</td>
</tr>
</tbody>
</table>
Parallel Tracking and Mapping (PTAM)

- monocular visual SLAM algorithm
- intended for small workspace AR (Augmented Reality) applications
- mapping and tracking are separated and run in two parallel threads
- mapping is based on keyframes
- new points are initialized with an epipolar search
- no feature or map uncertainties model (bundle adjustment on a vast number of image features)
- robust against partial camera occlusions (50 % of features available)
PTAM: what is a map

- a collection of $M$ map points and $N$ keyframes

- map point: a 3D point in the world \( p_{jw} = (x_{jw}, y_{jw}, z_{jw}, 1)^T \)
- keyframe: a pyramid of greyscale 8bpp images (i.e. 640 x 480, … , 80 x 60) and an associated camera-centred coordinate frame \( (K_i) \)
PTAM: map initialization

1. acquire the first keyframe
1. acquire the first keyframe

2. translate and rotate the camera while tracking the features (hyp. 10 cm translation)
PTAM: map initialization

1. acquire the first keyframe (triggered by user)

2. translate and rotate the camera while tracking the features (hyp. 10 cm translation)

3. acquire the second keyframe (user triggered) and build the map (epipolar search)
PTAM: tracking thread

new key frame

...real-time task!

TRACKING THREAD

Map available?

TRUE

apply camera motion model

track the map (projection+measure)

update camera pose

AssessTrackingQuality

Get lost?

TRUE

Relocaliser

FALSE

Need New Keyframe?

TRUE

To Map Maker

FALSE

...back to the top
## Map point search

Fixed - range image search around the point’s predicted image location

1. Map point in original keyframe (first spotted)

2. Fixed range patch extracted from original image
   \[(8 \times 8)\]

3. Affine warp of the patch based on motion model pose estimate

4. Projection on the current view based on the motion model pose estimate

5. Comparison with the current view
   (Sum of Squared Differences score)

Examples of affine warping
PTAM: mapping thread

…slow but accurate!

MAPPING THREAD
MapMaker::run()

New Keyframe?

Locally converged?

Globally converged?

Add keyframe to the map
Add map points via epipolar search
Set Global and Local convergence to FALSE

Check map points and trash bad ones

Local Bundle

Global Bundle

…back to the top
PTAM: monocular SLAM on a quadrotor?

**Some basic facts:**
- Hummingbird quadrotor equipped with a monocular camera facing downwards
- Multi-Robot-Integration Platform (MIP) implements quadrotor communication, wireless camera sensor interface and some other useful stuff

Passing camera frames from MIP to PTAM will do the trick...
MIP: an overview

A C++ software aimed to develop control and estimation robotics algorithms

- Good level of modularity
- Use of abstracted low-level interfaces
- Interface with 3D simulation environment (Player/Gazebo)

### MIP components

- **Main**
  - Main of the program. Here is created and launched the Scheduler

- **Baselib**
  - Basic library for general purpose and robotics functionalities, e.g. IP communication, pose class, matrix class, ...

- **Resources**
  - Class providing interface modules respect to the hardware or the MIP platform facilities, e.g. camera, quadrotor, keyboard, ...

- **Algorithms**
  - Class collection of robotics algorithms, e.g. Visual odometry algorithm, estimate (Kalman filtering)

- **Tasks**
  - High level robot activities that must be execute in parallel, glueing algorithms and resources, e.g. tracking, deployment, target navigation, mutual localization, entrapment, exploration, ...
MIP: an overview

**Configuration file**
- Resource 1 (options)
- Task 1 (options)

**TASKS**
- Task 1
  - Run func.
  - Exe. time
- Task N
  - Run func.
  - Exe. time

**RESOURCES**
- Resource 1 quadrotor
- Resource M camera

**LOADER**

**SCHEDULER**
- Task1 run()
- Task2 run()
- TaskN run()

**ALGORITHMS**
- Algorithm 1
- Algorithm K

**Execution cycle**
MIP: use with PTAM and quadrotor

- QUADROTOR (simulated)
  - Sensor data (e.g. IMU)
  - Control
- CLOCHE KEYBOARD HIT (pilot quadrotor via keyboard)
- VISUAL ODOMETRY (communicate with PTAM)
- UAV LOCALIZATION (through PTAM)
- PTAM

Legend:
- Tasks
- Resources
- Algorithms
MIP: use with PTAM and quadrotor

1. Start 3D simulation environment (Player/Gazebo)
MIP: use with PTAM and quadrotor

1. Start 3D simulation environment (Player/Gazebo)

2. Select the configuration file and run MIP
MIP: use with PTAM and quadrotor

1. Start 3D simulation environment (Player/Gazebo)

2. Select the configuration file and run MIP

3. Start PTAM and navigate

